Time Series as Images: Vision Transformer for Irregularly Sampled Time Series

Zekun Li, Shiyang Li, Xifeng Yan
University of California, Santa Barbara
{zekunli, shiyangli, xyan}@cs.ucsb.edu

Abstract

Irregularly sampled time series are becoming increasingly prevalent in various domains, especially in medical applications. Although different highly-customized methods have been proposed to tackle irregularity, how to effectively model their complicated dynamics and high sparsity is still an open problem. This paper studies the problem from a whole new perspective: transforming irregularly sampled time series into line graph images and adapting powerful vision transformers to perform time series classification in the same way as image classification. Our approach largely simplifies algorithm designs without assuming prior knowledge and can be potentially extended as a general-purpose framework. Despite its simplicity, we show that it substantially outperforms state-of-the-art specialized algorithms on several popular healthcare and human activity datasets. Our code and data are available at https://github.com/Leezekun/ViTST.

1 Introduction

Time series data are ubiquitous in a wide range of domains, including healthcare, finance, traffic and climate science. With the advances in deep learning architectures such as LSTM (Graves, 2012), Temporal Convolutional Network (TCN) (Lea et al., 2017), and Transformer (Vaswani et al., 2017), numerous algorithms have been developed for time series modeling. However, these methods typically assume fully observed data points at regular intervals and fixed-size numerical inputs. They cannot deal with irregularly sampled ones, a sequence of samples with irregular intervals between their observation times. To tackle this challenge, highly specialized models were developed, which require a considerable amount of prior knowledge in the model architecture choice and design (Marlin et al., 2012; Lipton et al., 2016; Che et al., 2018; Horn et al., 2020; Zhang et al., 2022a; Shukla & Marlin, 2020; Zhang et al., 2022b).

The recently emerging transformer-based vision models, most notably Vision Transformers (Dosovitskiy et al., 2020)[1], have demonstrated strong performance on various vision tasks such as image classification and object detection. In this paper, we raise a simple question: *Since pretrained vision transformers have exceeded humans in various image recognition tasks, can they "visually" capture temporal patterns in the visualized time series data?* To answer this question, we explore the following minimalist approach: transform the irregularly sampled multivariate time series into line graphs (Fig. 1), arrange these line graphs into a standard RGB image, and finetune a pretrained vision transformer to perceive the image and perform the classification task. We dub this approach **ViTST**, short for **Vi**sion **T**ime **S**eries **T**ransformer. The line graph images could encode two kinds of informative patterns in multivariate time series: (1) the temporal dynamics of each variable in its corresponding line graph; and (2) the correlation of variables across different line graphs. We assume that vision transformers can capture pattern (1) via modeling local patch interactions within a single time series line graph images and pattern (2) from global patch interactions across different line graphs.

Experimental results show that our approach ViTST outperforms previous SOTA results by 2.4 and 1.2 AUROC points on two irregularly sampled healthcare datasets P19 (Reyna et al., 2019) and

[1]In this paper, we refer to vision transformers as a type of pretrained vision models based on Transformer, including ViT (Dosovitskiy et al., 2020), Swin Transformer (Liu et al., 2021), DeiT (Touvron et al., 2021), to name a few.

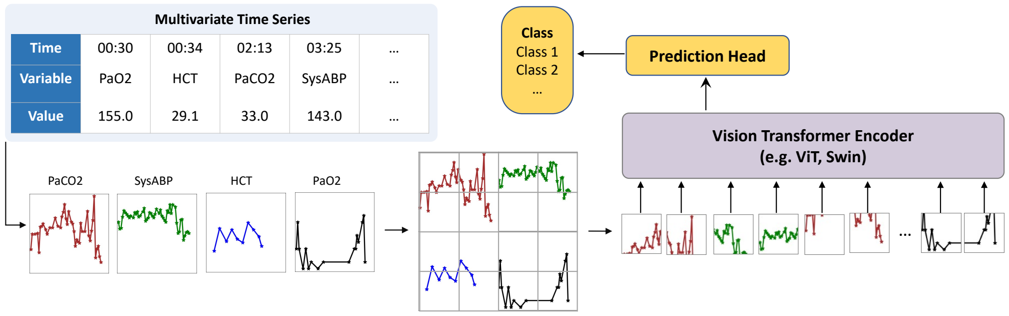


Figure 1: Illustration of our approach ViTST. The example is from a healthcare dataset P12 (Goldberger et al., 2000), which provides the irregularly sampled observations of 36 variables for patients (we only show 4 variables here for simplicity). Each column in the table is an observation of a variable, with the observed time and value. We plot separate line graphs for each variable and arrange them into an image, which is then fed into the vision transformer to perform the classification task.

P12 (Goldberger et al., 2000), and 6.8 F1 score points on a human activity dataset PAM (Reiss & Stricker, 2012). The results demonstrate the effectiveness of our approach, which is in contrast to its simplicity. ViTST can also be potentially extended as a general-purpose framework for time series modeling. It also attains excellent results compared with SOTA algorithms designed for regular time series modeling, demonstrating its generality. We believe this paper could open up a new direction and encourage the utilization of fast-evolving and well-studied computer vision techniques in the time series domain, such as better model architecture (Liu et al., 2022), data augmentation (Shorten & Khoshgoftaar, 2019), interpretability (Chefer et al., 2021), self-supervised learning (He et al., 2022), to name a few.

2 Approach

As illustrated in Fig. 1, ViTST consists of two steps: (1) transform multivariate time series into a concatenated line graph image; (2) utilize the vision transformer as an image classifier for the classification task. To begin with, we present some basic notations and problem formulation.

Notation. Let $\mathcal{D} = \{(\mathcal{S}_i, y_i)|i = 1, \cdots, N\}$ denote a time series dataset containing N samples. Every data sample is associated with a label $y_i \in \{1, \cdots, C\}$, where C is the number of classes. Each multivariate time series $\mathcal{S}_i$ consists of observations of D variables at most (some might have no observations). The observations for each variable d are given by a sequence of tuples with observed time and value $[(t_1^d, v_1^d), (t_2^d, v_2^d), \cdots, (t_{n_d}^d, v_{n_d}^d)]$, where n_d is the number of observations for variable d. If the intervals between observation times $[t_1^d, t_2^d, \cdots, t_{n_d}^d]$ are different across variables or samples, $\mathcal{S}_i$ is an irregularly sampled time series. Otherwise, it is regular time series.

Problem Formulation. Given the dataset $\mathcal{D} = \{(\mathcal{S}_i, y_i)|i = 1, \cdots, N\}$ containing N multivariate time series, we aim to predict the label $\hat{y}_i \in \{1, \cdots, C\}$ for each time series $\mathcal{S}_i$.

2.1 Time Series to Image Transformation

Time Series Line Graph. Time series line graph is a widely-used data visualization method to illustrate temporal data points at successive intervals. Each point on the line graph corresponds to an observation with an observed time and value. The horizontal axis is used to plot timestamps, and the vertical axis is used to plot values. Straight lines connect the points on the graph in the order of time, where the missing value interpolation is done automatically. We use markers "$\star$" to indicate the data point in the line. As the scale of different variables varies greatly, we plot the observations of each variable in an individual line graph, as shown in Fig. 1. The scales of each line graph $\mathrm{g}_{i,d}$ are kept the same across different time series $\mathcal{S}_i$. Different colors are used for each line graph to distinguish them.

Image Creation. Given a set of time series line graphs $\mathcal{G}_i = \{\mathrm{g}_1, \mathrm{g}_2, \cdots, \mathrm{g}_D\}$ for time series $\mathcal{S}_i$, we place them in a single image x_i using a pre-defined grid layout, in which the line graph of each variable is in a grid cell. Similar to (Fan et al., 2021), we experimentally found that a compact layout (*i.e.*, square grid) leads to consistently good performance. Specifically, given the D time series line graphs for a time series $\mathcal{S}_i$, we place them in a grid, whose size is $l \times l$ when $l \times (l-1) < D <= l \times l$, and $l \times (l+1)$ when $l \times l < D <= l \times (l+1)$. For example, there are 34, 36, and 17 variables in P19, P12, and PAM datasets, respectively. The default grid layouts are thus 6×6, 6×6, and 4×5. If the grid is not full, the cells at the end of the grid are left blank. More details on image creation and examples are provided in Appendix D.3.

2.2 Vision Transformers for Time Series Modeling

Given the image x_i transformed from time series $\mathcal{S}_i$, we leverage an image classifier to perceive the image and perform the classification task. The time series patterns in a line graph image involve both local (*i.e.*, the temporal dynamics of a single variable in a line graph) and global (the correlation among variables across different line graphs) contexts. To better capture these patterns, we choose the recently developed vision transformers, which show strong abilities to capture local and global dependencies (Dosovitskiy et al., 2020; Liu et al., 2021).

Preliminary. Vision Transformer (ViT) (Dosovitskiy et al., 2020) is originally adapted from NLP. An image is split into fix-sized patches, each linearly embedded and augmented with position embeddings. The resulting sequence of vectors is fed into a standard Transformer encoder to obtain patch representations. An extra classification token is added to the sequence to perform classification or other tasks. ViT models *global* inter-unit interactions between each pair of patches, which faces efficiency issues when dealing with high-resolution images.

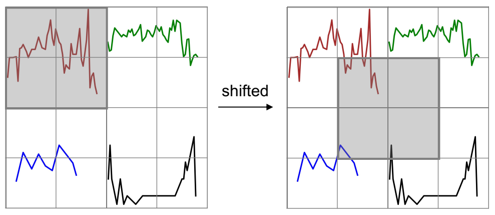


Figure 2: Illustration of the shifted window approach of Swin Transformer. The self-attention is calculated within each window (grey box). When the window is within a single line graph, the local interactions are captured. After shifting, the window contains patches from different line graphs, and thus global cross-variable interactions are modeled.

Swin Transformer, on the other hand, has a hierarchical architecture that contains multi-level feature maps and computes self-attention locally within non-overlapping windows, significantly reducing the computation complexity and improving the recognition performance. As illustrated in Fig. 2, the self-attention is calculated within each non-overlapping window. When the sliding window is within a single line graph for variable d, the local intra-variable interactions and temporal dynamics of the variable d are captured. The shifted window block SW-MSA enables the connection of different windows. After shifting, the window spans across different line graphs. With multiple stages of blocks, the global interactions among the patches from all the line graphs can be modeled, and thus the correlation between different variables is learned. We evaluate our idea using ViT and Swin in this work and use Swin as the default backbone vision model if not specified. Note that any other vision model can be applied under this framework.

Inference. We use the vision transformers to predict the labels of time series in the same way as image classification. The outputs of Swin Transformer blocks at the final stage are used as the patch representations, upon which a flattened layer with a linear head is applied to obtain the prediction $\hat{y}_i$. As for ViT, the representation of the additional classification token at the final layer is used for prediction. We use the cross-entropy loss when fine-tuning the model on the classification task.

3 Experiments

3.1 Experimental Setup

Datasets and Metrics. We conduct experiments using two popular healthcare datasets P19 and P12, and one human activity dataset PAM (Reiss & Stricker, 2012). We used the processed data provided by Raindrop (Zhang et al., 2022a)[2]. More details are given in Appendix B.1. We employed the same five data splits for all comparison baselines, as provided. The evaluation metrics were consistent across all experiments, including the Area Under a ROC Curve (AUROC) and Area Under

[2]https://github.com/mims-harvard/Raindrop

Table 1: Comparison with the baseline methods on irregularly sampled time series classification task. **Bold** indicates the best performer, while underline represents the second best.

Methods	P19		P12		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1 score
Transformer	80.7 ± 3.8	42.7 ± 7.7	83.3 ± 0.7	47.9 ± 3.6	83.5 ± 1.5	84.8 ± 1.5	86.0 ± 1.2	85.0 ± 1.3
Trans-mean	83.7 ± 1.8	45.8 ± 3.2	82.6 ± 2.0	46.3 ± 4.0	83.7 ± 2.3	84.9 ± 2.6	86.4 ± 2.1	85.1 ± 2.4
GRU-D	83.9 ±1.7	46.9 ± 2.1	81.9 ± 2.1	46.1 ± 4.7	83.3 ± 1.6	84.6 ± 1.2	85.2 ± 1.6	84.8 ± 1.2
SeFT	81.2 ± 2.3	41.9 ± 3.1	73.9 ± 2.5	31.1 ± 4.1	67.1 ± 2.2	70.0 ± 2.4	68.2 ± 1.5	68.5 ± 1.8
mTAND	84.4 ± 1.3	50.6 ± 2.0	84.2 ± 0.8	$\underline{48.2}$ ± 3.4	74.6 ± 4.3	74.3 ± 4.0	79.5 ± 2.8	76.8 ± 3.4
IP-Net	84.6 ± 1.3	38.1 ± 3.7	82.6 ± 1.4	47.6 ± 3.1	74.3 ± 3.8	75.6 ± 2.1	77.9 ± 2.2	76.6 ± 2.8
DGM^2-O	86.7 ± 3.4	44.7 ± 11.7	84.4 ± 1.6	47.3 ± 3.6	82.4 ± 2.3	85.2 ± 1.2	83.9 ± 2.3	84.3 ± 1.8
MTGNN	81.9 ± 6.2	39.9 ± 8.9	74.4 ± 6.7	35.5 ± 6.0	83.4 ± 1.9	85.2 ± 1.7	86.1 ± 1.9	85.9 ± 2.4
Raindrop	87.0 ± 2.3	$\underline{51.8}$ ± 5.5	82.8 ± 1.7	44.0 ± 3.0	88.5 ± 1.5	89.9 ± 1.5	89.9 ± 1.5	89.8 ± 1.0
VITST-ViT	$\underline{87.9}$ ± 2.5	51.6 ± 3.7	$\underline{84.8}$ ± 1.3	48.1 ± 3.8	$\underline{93.4}$ ± 0.7	$\underline{94.7}$ ± 0.9	$\underline{94.1}$ ± 0.7	$\underline{94.3}$ ± 0.7
VITST-Swin	**89.4** ± 1.9	**52.8** ± 3.8	**85.6** ± 1.1	**49.8** ± 2.5	**96.1** ± 0.7	**96.8** ± 1.1	**96.5** ± 0.7	**96.6** ± 0.9

Precision-Recall Curve (AUPRC) for the imbalanced datasets P12 and P19. For the PAM dataset, which has a more balanced class distribution, we reported Accuracy, Precision, Recall, and F1 score.

Baselines. We compare our approach with several state-of-the-art methods specialized for irregularly sampled time series: Transformer (Vaswani et al., 2017), Trans-mean (Transformer with an imputation method that replaces the missing value with the average observed value of the variable), GRU-D (Che et al., 2018), SeFT (Horn et al., 2020), mTAND (Shukla & Marlin, 2020), IP-Net (Shukla & Marlin, 2018), and Raindrop (Zhang et al., 2022a). Raindrop is the best-performing method before this work. Besides, two methods initially designed for forecasting tasks are also compared, including DGM^2-O (Wu et al., 2021b) and MTGNN (Wu et al., 2020). The implementations and hyperparameter settings of these baselines all follow Raindrop (Zhang et al., 2022a). The performances are averaged on 5 different data splits, which are kept the same across all the compared methods.

Implementation. We use the Matplotlib package to draw the line graphs and save them as standard RGB images. The grid layouts of data in P19, P12, and PAM dataset are 6×6, 6×6, and 4×5. We set the size of each grid cell (line graph) as 64×64, and thus the image sizes are 384×384, 384×384, and 256×320, respectively. One can also directly set the image size to any size, regardless of the grid cell size. We use the checkpoints of ViT[3] and Swin Transformer[4] pre-trained on the ImageNet-21K dataset.

3.2 Main Results

As seen from Table 1, under our framework, ViT and Swin Transformer both outperform the specialized state-of-the-art algorithms on all these three datasets, which suggests the effectiveness of our proposed framework that utilizes vision transformers for time series modeling. On the P19 and P12 datasets, ViTST-Swin improves the state-of-the-art results by 2.4 and 1.2 AUROC points, respectively. As for the PAM dataset, the improvement is even more significant: 7.6 points in Accuracy, 6.9 points in Precision, 6.6 points in Recall, and 6.8 points in F1 score. The larger improvement on the PAM dataset might be due to the lower missing ratio of PAM (60.0%) than P19 (94.9%) and P12 (88.4%), meaning that there are more observed values to better recover the fully observed line graphs and reflect patterns (see Fig. 3 for created images from these three datasets).

4 Conclusion and Future Work

In this paper, we introduced a new perspective for irregularly sample multivariate time series modeling by transforming them into images, which enables the use of powerful vision transformers. This approach is simple and general since any type of time series can be transformed into line graph images and handled. Despite its simplicity, our approach demonstrates strong performance against highly specialized state-of-the-art methods on several popular healthcare datasets. Our approach can also be potentially extended as a general-purpose framework for various time series tasks, which we leave for future work. We believe this work could open up a new direction and encourage the reuse of fast-evolving computer vision techniques in the time series modeling domain.

[3]https://huggingface.co/google/vit-base-patch16-224-21k

[4]https://huggingface.co/microsoft/swin-base-patch4-window7-224-in22k

REFERENCES

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 782–791, 2021.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Shashank Rajput, Michael Gira, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *arXiv preprint arXiv:2206.06565*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Quanfu Fan, Chun-Fu Chen, and Rameswar Panda. Can an image classifier suffice for action recognition? In *International Conference on Learning Representations*, 2021.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

Nima Hatami, Yann Gavet, and Johan Debayle. Classification of time-series images using deep convolutional neural networks. In *Tenth international conference on machine vision (ICMV 2017)*, volume 10696, pp. 242–249. SPIE, 2018.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, June 2022.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *International Conference on Machine Learning*, pp. 4353–4363. PMLR, 2020.

Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Zachary C Lipton, David Kale, and Randall Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine learning for healthcare conference*, pp. 253–270. PMLR, 2016.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022.

Benjamin M Marlin, David C Kale, Robinder G Khemani, and Randall C Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT international health informatics symposium*, pp. 389–398, 2012.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Predicting healthcare trajectories from medical records: A deep learning approach. *Journal of biomedical informatics*, 69:218–229, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers*, pp. 108–109. IEEE, 2012.

Matthew A Reyna, Chris Josef, Salman Seyedi, Russell Jeter, Supreeth P Shashikumar, M Brandon Westover, Ashish Sharma, Shamim Nemati, and Gari D Clifford. Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. In *2019 Computing in Cardiology (CinC)*, pp. Page–1. IEEE, 2019.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Satya Narayan Shukla and Benjamin Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2018.

Satya Narayan Shukla and Benjamin Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2020.

Srijan Sood, Zhen Zeng, Naftali Cohen, Tucker Balch, and Manuela Veloso. Visual time series forecasting: an image-driven approach. In *Proceedings of the Second ACM International Conference on AI in Finance*, pp. 1–9, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

RK Tripathy and U Rajendra Acharya. Use of features from rr-time series and eeg signals for automated classification of sleep stages in deep neural network framework. *Biocybernetics and Biomedical Engineering*, 38(4):890–902, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015a.

Zhiguang Wang and Tim Oates. Spatially encoding temporal correlations to classify temporal data using convolutional neural networks. *arXiv preprint arXiv:1509.07481*, 2015b.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021a.

Yinjun Wu, Jingchao Ni, Wei Cheng, Bo Zong, Dongjin Song, Zhengzhang Chen, Yanchi Liu, Xuchao Zhang, Haifeng Chen, and Susan B Davidson. Dynamic gaussian mixture based deep generative model for robust forecasting on sparse multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 651–659, 2021b.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 753–763, 2020.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

Hao Xue and Flora D Salim. Prompt-based time series forecasting: A new task and dataset.

Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Multi-directional recurrent neural networks: A novel method for estimating missing data. In *Time series workshop in international conference on machine learning*, 2017.

George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.

Xiang Zhang, Marko Zeman, Theodoros Tsiligkaridis, and Marinka Zitnik. Graph-guided network for irregularly sampled multivariate time series. In *International Conference on Learning Representations*, 2022a.

Xinlu Zhang, SHIYANG LI, Zhiyu Chen, Xifeng Yan, and Linda Petzold. Improving medical predictions by irregular multimodal electronic health records modeling. *ArXiv*, abs/2210.12156, 2022b.

Yuan Zhang. Attain: Attention-based time-aware lstm networks for disease progression modeling. In *In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-2019), pp. 4369-4375, Macao, China.*, 2019.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *arXiv preprint arXiv:2201.12740*, 2022.

APPENDIX

A RELATED WORK

Irregularly Sampled Time Series. An irregularly sampled time series is a sequence of observations with irregular intervals between observation times. In a multivariate setting, different variables within the same time series may not align. Such characteristics have posed a significant challenge to the standard time series modeling methods, which typically assume fully observed and regularly sampled data points. A common approach to handle irregular sampling is to convert continuous-time observations into fixed time intervals (Marlin et al., 2012; Lipton et al., 2016). To incorporate the dynamics between observations, GRU-D (Che et al., 2018) decays the hidden states based on gated recurrent units (GRU) (Chung et al., 2014), which takes as input the observations' values and also times. (Pham et al., 2017) modified the forget gate of LSTM (Graves, 2012) to better account for irregularity. Similarly, (Yoon et al., 2017) proposed an approach based on multi-directional RNN, which can capture the inter- and intra-steam patterns. Besides the recurrent and differential equation-based model architectures, recent work has explored attention-based models. Transformer (Vaswani et al., 2017) is naturally able to handle arbitrary sequences of observations. ATTAIN (Zhang, 2019) incorporates attention mechanism with LSTM to model the time irregularity between observations. SeFT (Horn et al., 2020) maps the irregular time series into a set of observations based on differentiable set functions and utilizes an attention mechanism for classification. mTAND (Shukla & Marlin, 2020) presented a multi-time attention network, which learns continuous-time embeddings coupled with a multi-time attention mechanism to deal with the continuous-time inputs. UTDE (Zhang et al., 2022b) integrated embeddings from mTAND and classical imputed time series with learnable gates to take their advantages for tackling complex temporal patterns. Raindrop (Zhang et al., 2022a) modeled the irregularly sampled time series as graphs and utilized graph neural networks to model the relationships between different variables. Overall, these methods are all highly specialized for irregular time series. In this work, we explore a simple and general vision transformer-based approach for irregularly sampled time series modeling without using dedicated model architecture modifications.

Numerical Time Series Modeling Methods with Transformer. Transformers possess superior abilities to capture long-range dependencies in sequential data, making them appealing to time series modeling (Li et al., 2019). A surge of transformer-based methods have been proposed and successfully applied to various time series modeling tasks, such as forecasting (Li et al., 2019; Zhou et al., 2021; Wu et al., 2021a; Zhou et al., 2022), classification (Zerveas et al., 2021), and anomaly detection (Xu et al., 2021). These methods are usually designed for regular time series settings, where they view multivariate numerical values at the same timestamp as a unit and model temporal interactions across different units. A recent work (Nie et al., 2022), on the other hand, segments each univariate time series into a sequence of sub-series and models their interactions independently.

Time Series as Other Modalities. The recently emerging pre-trained transformer-based models, initially proposed in Natural Language Processing (NLP) field, have since come to monopolize the state-of-the-art performance across various downstream tasks in NLP and Computer Vision (CV) fields. For example, the pre-trained language model BERT (Devlin et al., 2018) and GPTs (Radford et al., 2018; 2019; Brown et al., 2020) can be adapted to various NLP tasks. Some non-language tasks can also be solved by these pre-trained transformer-based language models by transforming them into language sentence prompts (Dinh et al., 2022). A recent work (Xue & Salim) tried to represent the time series in natural language and utilize pre-trained language models to forecast. However, such a method has difficulties modeling long-range multivariate time series as it usually involves tens of thousands of numerical values, which cannot be fitted into the language models (512/1024 max tokens for most LMs). In addition, it is hard to express the informative irregularity of time series in natural language sentences. By contrast, we transform numerical time series data into images and utilize pre-trained transformer-based vision models to perform time series modeling, which doesn't the issues. Note that some prior studies tried to transform time series into Gramian fields (Wang & Oates, 2015a), recurring plots (Hatami et al., 2018; Tripathy & Acharya, 2018), and Markov transition fields (Wang & Oates, 2015b) images and utilize CNN to perform classifications. However, these methods are not domain-agnostic and require domain knowledge in designing specialized imaging methods. Another related work (Sood et al., 2021) employed convolutional autoencoders to complete the image converted from time series for forecasting purposes. Nevertheless, this method is less effective than numerical techniques and cannot handle irregularly sampled time series data. By contrast, our method transforms time series into line graph RGB images without assuming prior knowledge and achieves better results than the highly specialized methods.

B Experiment Details

Table 2: Statistics of the irregularly sampled time series datasets (Zhang et al., 2022a). "#Avg. obs." denotes the average number of observations for each sample. "Static info" indicates if the time series sample is associated with static attributes (*e.g.*, genders).

Datasets	#Samples	#Variables	#Avg. obs.	#Classes	Static info	Imbalanced	Missing ratio
P19	38,803	34	401	2	True	True	94.9%
P12	11,988	36	233	2	True	True	88.4%
PAM	5,333	17	4,048	8	False	False	60.0%

B.1 Datasets

We conduct experiments using three popular datasets in healthcare and human activity, which are processed by (Zhang et al., 2022a). The statistics is shown in Table 2.

P19: PhysioNet Sepsis Early Prediction Challenge 2019. [5] P19 dataset (Reyna et al., 2019) contains the clinical data of 38,803 patients, and the goal is to predict the occurrence of sepsis within the next 6 hours. Each patient is monitored by 34 irregularly sampled sensors with 8 vital signs and 26 laboratory values. 6 demographic information is also provided. This is a binary classification task, and the dataset is highly imbalanced with around 4% positive samples. The missing ratio is 94.9%.

P12: PhysioNet Mortality Prediction Challenge 2012. [6] P12 dataset (Goldberger et al., 2000) includes the clinical data of 11,988 ICU patients (12 inappropriate patient samples are removed). 36 irregularly sampled sensor observations and 6 static demographics features of each patient are provided. The goal is to predict the mortality of patients (binary classification). This dataset is also highly imbalanced with around 86% negative samples. The missing ratio is 88.4%.

PAM: PAMAP2 Physical Activity Monitoring. [7] The original PAM dataset contains data of 18 physical activities with 9 subjects wearing 3 inertial measurement units. To make it suitable for irregular time series classification, (Zhang et al., 2022a) excluded the ninth subject due to its short length of sensor readouts. The continuous signals are segmented with a time window size of 600 and an overlapping rate of 50%. 10 out of the 18 activities in the original dataset are excluded as they are associated with less than 500 samples, and 8 activities remain. Therefore, the task is an 8-way classification. Finally, there are 5,333 samples, each with 600 continuous observations. To simulate the irregular time series setting, 60% of the observations are randomly removed. No static features are provided, and the 8 categories are approximately balanced. The missing ratio is 60.0%.

B.2 Implementation and Training

Image Creation. The time-series-to-image transformation can be implemented using the Matplotlib package[8] with the following few lines of code.

```python
def TS2Image(t, v, D, colors, image_height, image_width, grid_height, grid_width):
    import matplotlib.pyplot as plt
    plt.figure(figsize=(image_height/100, image_width/100), dpi=100)
    for d in range(D): # enumerate the multiple variables
        plt.subplot(grid_height, grid_width, d+1) # position in the grid
        # plot line graph of variable d
        plt.plot(t[d], v[d], color=colors[d], linestyle="-", marker="*")
```

[5] https://physionet.org/content/challenge-2019/1.0.0/

[6] https://physionet.org/content/challenge-2012/1.0.0/

[7] https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring

[8] https://matplotlib.org/

As mentioned in Section 2.1, the observations of each variable are plotted in a separate line graph. We use straight lines "–" to connect consecutive points on the line graph. To distinguish the observed data points from the line, we use markers "*". To differentiate these line graphs, we use variable-specific colors for each line graph.

Table 3: Templates for transforming static features to natural language sentences.

Dataset	Static features	Template	Example
P19	*Age*, *Gender*, *Unit1 (medical ICU)*, *Unit2 (surgery ICU)*, *HospAdmTime*; *ICULOS (ICU length-of-stay)*	A patient is {*Age*} years old, {*Gender*}, went to {*Unit1&Unit2*} {*HospAdmTime*} hours after hospital admit, had stayed there for {*ICULOS*} hours.	A patient is 65 years old, female, went to the medical ICU 10 hours after hospital admit, had stayed there for 20 hours.
P12	*RecordID*, *Age*, *Gender*, *Height* (cm), *ICUType*, *Weight* (kg)	A patient is {*Age*} years old, {*Gender*}, {*Height*} cm, {*Weight*} kg, stayed in {*ICUType*}.	A patient is 48 years old, male, 171 cm, 78 kg, stayed in surgical ICU.

Incorporating static features. The P12 and P19 datasets provide patients' demographics, such as weight, height, and ICU type. This static information will not change over time and can be well described by the natural language. To incorporate them into our framework, we transform them into natural language sentences via templates as shown in Table 3 and utilize the pre-trained Roberta-base[9] (Liu et al., 2019) to obtain textual features, which is concatenated with the visual feature from vision transformer to perform classification. Note that the static feature is also applied to all the baselines we compare.

Training. We apply the cutout (DeVries & Taylor, 2017) augmentation method on the input images from P12 and P19 datasets during training to avoid over-fitting caused by upsampling. Specifically, 16 square regions with 16×16 size are randomly masked in each image. The models are trained using A6000 GPUs with 48G memory. As P12 and P19 datasets are highly imbalanced, we upsample the minority class to the same size as the majority class. We fine-tune Swin Transformer 2 and 4 epochs on upsampled P19 and P12 datasets and 20 epochs on the PAM dataset. The batch sizes are 48 for P19 and P12, and 72 for PAM. The learning rate is 2e-5. As for the compared baselines, we follow the implementations and hyperparameter settings in Raindrop (Zhang et al., 2022a): The batch size is 128, and all the compared models are trained for 20 epochs. As the P12 and P19 datasets are highly imbalanced, we make each batch balanced with half negative and half positive samples.

C Visualization

To understand what patterns ViTST capture in the time series line graph images, we presented the averaged attention map of a ViTST-ViT in Fig. 3. As can be seen, the model learns to attend to the lines instead of the whitespace. In addition, we observe that the model correctly focuses on observed data points (dots) and changing slopes on the lines, which indicates the observation and trend information. Some flat line graphs which don't reflect many dynamic patterns receive less attention.

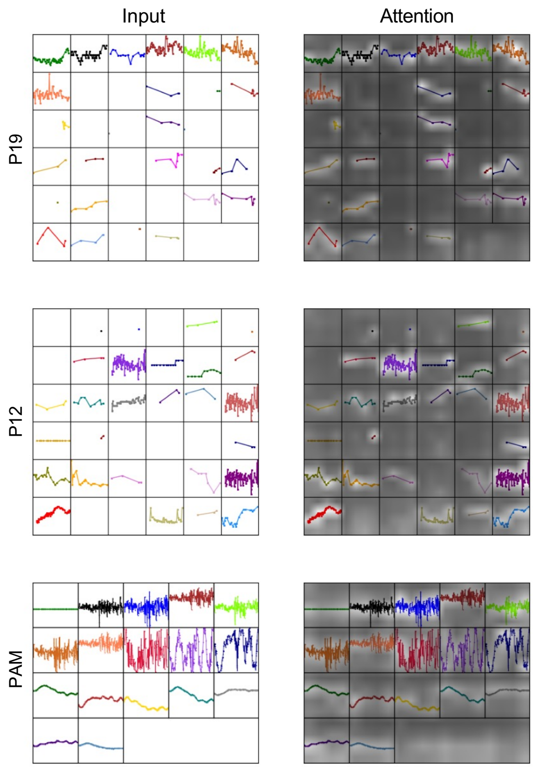
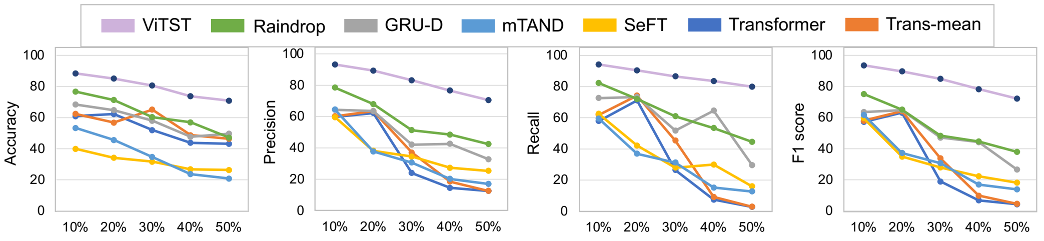


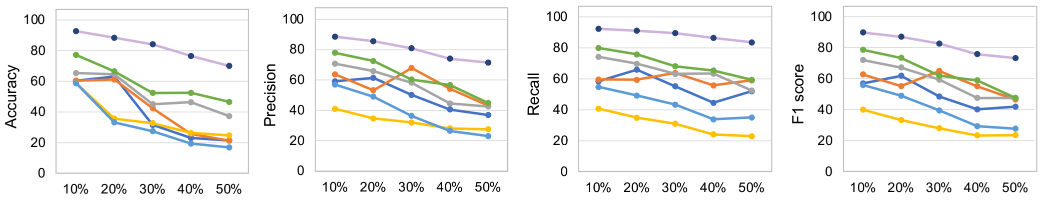
Figure 3: Illustration of the averaged attention map of ViTST on three images from P19, P12, and PAM datasets, respectively. Left: input images. Right: attention maps.

[9]https://huggingface.co/roberta-base

D Additional Experiments



(a) Leave-**fixed**-sensors-out



(b) Leave-**random**-sensors-out

Figure 4: Performance in leave-**fixed**-sensors-out and leave-**random**-sensors-out settings on PAM dataset. The x-axis is the "missing ratio" which denotes the ratio of masked variables. Detailed numbers are provided in Table 12.

D.1 Leaving-sensors-out Settings.

We further evaluate models' performance in more challenging leave-sensors-out settings, where the observations of a subset of sensors (variables) are masked during testing. Following (Zhang et al., 2022a), we experiment with two setups on PAM dataset: (1) *leave-**fixed**-sensors-out*, which drops a fixed set of sensors across all the samples and compared methods; (2) *leave-**random**-sensors-out* which drops the sensors randomly. Only the observations in the validation and test set are dropped. The training set is kept unchanged. For a fair comparison, we dropped the same set of sensors in the *leave-**fixed**-sensors-out* setting as in (Zhang et al., 2022a).

The results are presented in Fig. 4, from which we observe that our approach consistently achieves the best performance and outperforms the compared baselines by a large margin. With the missing ratio ranging from 10% to 50%, the performance improvement over the previous best model becomes increasingly significant. When half of the variables are dropped, our approach can still achieve acceptable performance, exceeding the best-performed baseline by up to 50.2% in Accuracy, 40.7% in Precision, 59.6% in Recall, and 54.0% in the F1 score, which suggests the robustness of our approach to missing observations in time series.

D.2 Backbone Vision Models

We tested the performance of different backbone vision models under our framework. In addition to the transformer-based ViT and Swin Transformer, we tested a CNN-based model ResNet[10]. We also report the performance of Swin Transformer trained from scratch and Raindrop for comparison. The results are presented in Fig. 5. The pre-trained ViT performs similarly to Swin Transformer. They both outperform the previous state-of-the-art method Raindrop, which suggests the effectiveness of our proposed framework that utilizes vision transformers for time series modeling. However, the CNN-based ResNet achieves much worse performance than the transformer-based models Swin Transformer and ViT, showing that our framework's superior performance derives not only from the idea of casting time series classification to image classification but also the strong image recognition ability of vision transformers. Swin Transformer trained from scratch underperforms its pre-trained counterpart by a large margin, which shows that knowledge obtained from pre-training on natural images could contribute to recognizing patterns in synthetic time series line graph images. It also

[10]https://huggingface.co/microsoft/resnet-50

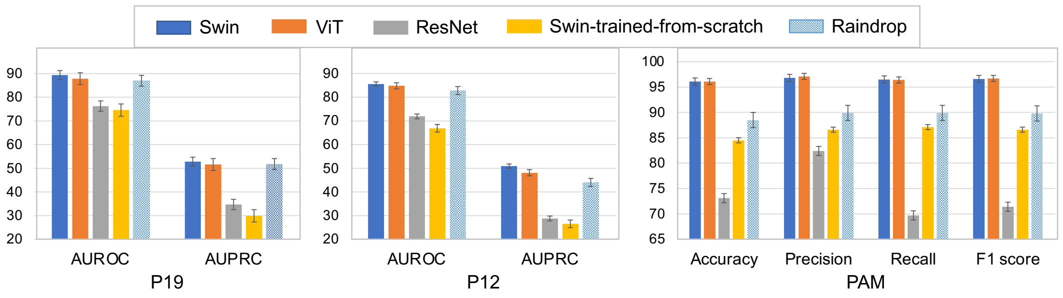


Figure 5: Performance of different backbone vision models and the state-of-the-art model Raindrop on P19, P12, and PAM datasets. Detailed numbers are provided in Table 13 in Appendix D.6.

reveals the advantages of our proposed framework: pre-trained vision models can be easily leveraged for time series modeling.

D.3 Time Series Line Graph Image Creation

Table 4: Ablation studies on different designs when drawing the time series line graphs.

Methods	P19		P12		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1 score
ViTST	89.4 ± 1.9	52.8 ± 3.8	85.6 ± 1.1	49.8 ± 2.5	96.1 ± 0.7	96.8 ± 1.1	96.5 ± 0.7	96.6 ± 0.9
w/o interpolation	87.5 ± 1.5	51.2 ± 3.6	84.1 ± 1.4	48.3 ± 3.5	96.0 ± 1.1	96.8 ± 0.9	96.4 ± 0.9	96.6 ± 0.9
w/o markers	88.3 ± 1.6	51.0 ± 2.4	84.8 ± 1.3	48.7 ± 3.8	94.1 ± 0.9	95.1 ± 0.7	94.8 ± 1.1	94.9 ± 0.8
w/o colors	85.3 ± 0.8	48.5 ± 2.1	83.9 ± 1.1	46.5 ± 3.2	92.9 ± 1.9	94.9 ± 1.2	93.6 ± 1.5	94.1 ± 1.5

As mentioned in Section 2.1, there are several key designs in drawing the line graphs for irregularly sampled multivariate time series and creating the images: (1) the linear *interpolation*, *i.e.*, linking the consecutive observed data points on the line graphs; (2) *markers* for observed data points to distinguish them from the "interpolated" ones on the line graph; (3) variable-specific line *colors* to distinguish different line graphs. We conducted ablation studies to test their effectiveness. The results are presented in Table 4. We can see that the performance decreases without either of these designs. However, the performance drop of removing interpolation and markers are not as significant as removing variable-specific line colors, which is reasonable as it is most observable on the line graph images and distinguishes different line graphs.

In addition to the above-mentioned designs, we have tried different strategies from the following perspectives when creating the time series line graph images.

Table 5: Ablation study on different strategies to decide the line graph limit. The default strategy is directly set the axis limit as the range of all observed values on the dataset. "IQR", "SD", and "MZS' denote three strategies to remove extreme value, *i.e.,* Interqurtile Range, Standard Deviation, and Modified Z-score. The reported numbers are averaged on 5 data splits.

Strategies	P19		P12		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1 score
Default	**89.4** ± 1.9	**52.8** ± 3.8	**85.6** ± 1.1	**49.8** ± 2.5	96.1 ± 0.7	96.8 ± 1.1	96.5 ± 0.7	96.6 ± 0.9
IQR	88.2 ± 0.8	49.6 ± 1.7	84.5 ± 1.1	48.9 ± 2.6	95.9 ± 0.7	96.8 ± 0.7	96.1 ± 0.7	96.4 ± 0.7
SD	87.4 ± 1.6	51.2 ± 3.6	84.6 ± 1.7	47.1 ± 2.9	**96.6** ± 0.9	**97.1** ± 0.8	**97.0** ± 0.6	**97.0** ± 0.7
MZS	87.3 ± 1.0	50.8 ± 3.7	84.3 ± 1.4	47.1 ± 2.1	96.0 ± 1.1	96.8 ± 0.9	96.4 ± 0.9	96.6 ± 0.9

Axis Limits of Line Graphs. The axis limits determine the plot area of the line graphs and the range of displayed timestamps and values. The default strategy is to directly set the limits of x-axis and y-axis as the ranges of all the observed timestamps and values across the dataset. However, we notice that there exist some extreme observed values for some variables, making the the range of y-axis very large. As a consequence, most plotted points of observations clusters in a small area and the drawn line graphs are flat (see Fig. 6). Note that the widely used linear normalization and standardization methods will not make a difference on the created images, as the relative magnitudes keep unchanged. We thus tried the following strategies to remove extreme values and narrow the range of y-axis:

- Interquartile Range (IQR): IQR is one of the most extensively used methods for outlier detection and removal. The interquartile range is calculated based on the first and third quartiles of all the observed values of each variable in the dataset and then used to calculate the upper and lower limits.
- Standard Deviation (SD): The upper and lower boundaries are calculated by taking 3 standard deviations from the mean of observed values for each variable across the dataset. This method usually assumes the data is normally distributed.
- Modified Z-score (MZ): A z-score measures how many standard deviations away a value is from the mean and is similar to the standard deviation method to detect outliers. However, z-scores can be influenced by extreme values, which modified z-scores can better handle. We set the upper and lower limits as the values whose modified z-scores are 3.5 and -3.5.

Using these methods to narrow the range of y-axis means some extreme values will be removed and not displayed in the plotted line graph. The comparison of model performance trained on images created with different strategies is shown in Table 5. We observe that these three methods that remove extreme values hurt the performance, except SD on PAM dataset. Although they narrow the value range and highlight the dynamic patterns of line graphs, they discard the extreme values which might be informative themselves. This observation suggests that our approach might not need data preprocessing on the time series, which further proves its advantage in simplicity.

Table 6: Ablation study on grid layouts and image sizes on P19.

Grid Layout	Image Size	AUROC	AUPRC
4×9	256×576	87.4 ± 1.9	48.1 ± 4.5
5×7	320×448	87.9 ± 1.9	49.6 ± 2.7
6×6	384×384	**89.4** ± 1.9	**52.8** ± 3.8
6×6	224×224	88.7 ± 1.4	52.3 ± 0.6

Table 7: Ablation study on grid layouts and image sizes on P12.

Grid Layout	Image Size	AUROC	AUPRC
4×9	256×576	84.0 ± 1.4	47.9 ± 2.6
5×8	320×512	84.1 ± 1.6	47.2 ± 2.3
6×6	384×384	**85.6** ± 1.1	**49.8** ± 2.5
6×6	224×224	85.2 ± 2.1	48.8 ± 3.7

Table 8: Ablation study on grid layouts and image sizes on PAM.

Grid Layout	Image Size	Accuracy	Precision	Recall	F1 score
2×9	128×576	95.9 ± 1.4	96.5 ± 1.0	**95.9** ± 1.2	96.0 ± 0.5
3×6	192×384	**96.1** ± 0.8	96.7 ± 0.5	**95.9** ± 0.9	96.2 ± 0.7
4×5	256×320	**96.1** ± 0.7	**96.8** ± 1.1	96.5 ± 0.7	**96.6** ± 0.9
4×5	224×224	95.9 ± 0.6	96.7 ± 0.8	**95.9** ± 0.6	96.3 ± 0.7

Grid Layout and Image Size. We conducted experiments to investigate the influence of grid layouts and image sizes on performance. Specifically, for a fair comparison of different grid layouts, we fixed the size of each grid cell as 64×64 and alter the grid layouts. The results on P19, P12, and PAM datasets are listed in Table 6, Table 7, and Table 8, respectively. As can be seen, the square grid layouts achieve consistently good results on three datasets. We conjecture that this is because the square layout ensures that the distance between any two line graphs is shortest. We also tried directly setting the image size as standard 224×224, and found the performance differences are marginal, showing that our method is robust to various image sizes.

Order of Line Graphs. We place the line graphs of each variable in a pre-defined grid layout to form the image. If the grid is not full, the empty cells at the end will be left blank. We empirically found that ViTST-Swin will learn to pay less attention to this empty region. We thus sort variables according to their missing ratios on the whole training set and keep the order fixed across different samples, ensuring the sparse line graphs of variables with few observations are placed at the end, next to the padded empty cells. We hypothesize such an order may facilitate the model to ignore the line graphs with the least observations instead of the informative line graphs with more observations if there

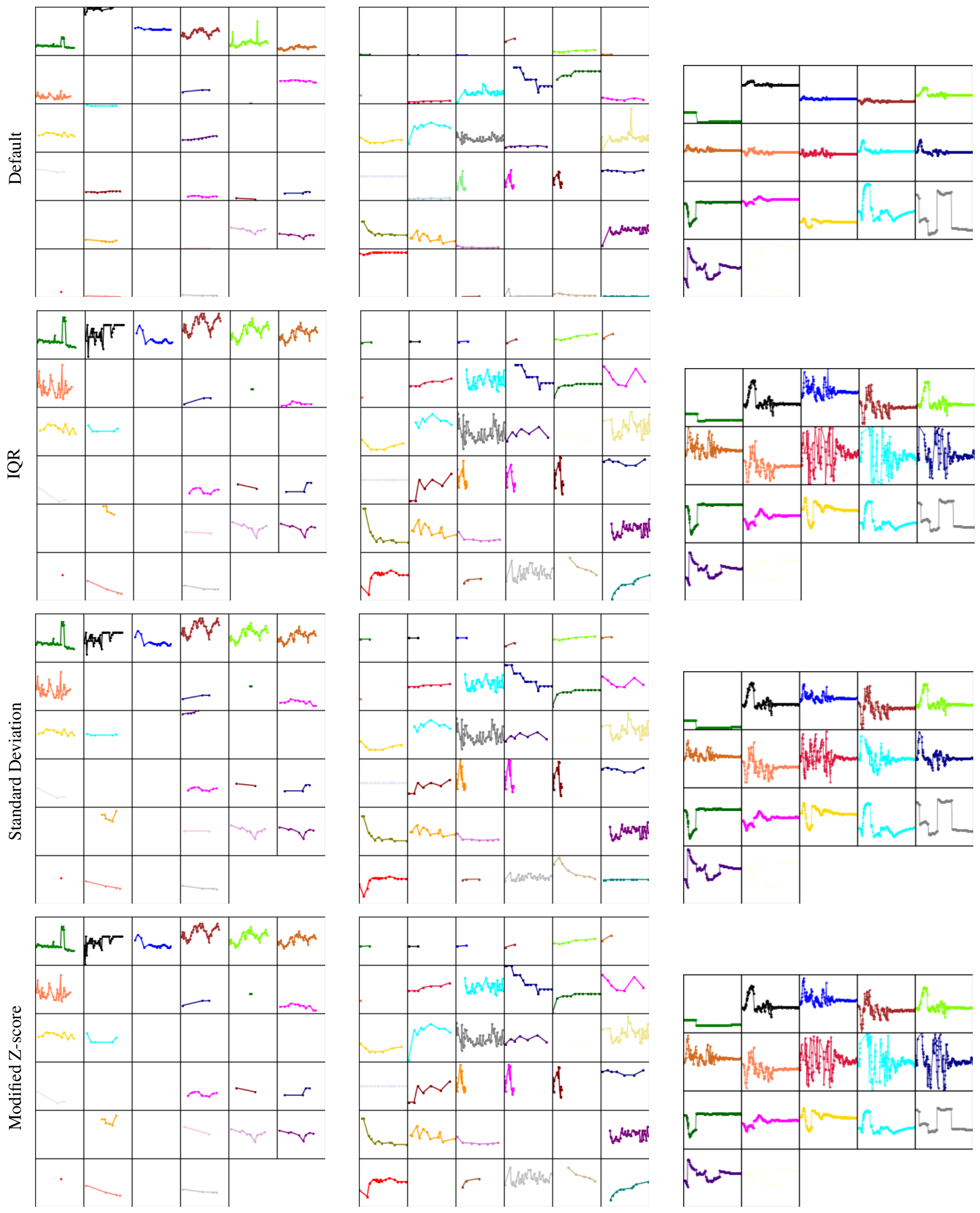


Figure 6: The images created with different strategies for three samples from P19, P12, and PAM dataset, respectively (sample "p000019" for P19, "132548" for P12, and "0" for PAM).

Table 9: Ablation studies on variables orders on P19 and P12 datasets.

Sorted	P19		P12	
	AUROC	AUPRC	AUROC	AUPRC
✗	88.3 ± 1.5	51.3 ± 3.0	85.6 ± 1.1	49.8 ± 2.5
✓	89.4 ± 1.9	52.8 ± 3.8	84.9 ± 1.8	49.8 ± 4.2

exists the ignorance bias caused by padded empty cells. To verify this, we trained a ViTST model on P19 and P12 dataset with random and sorted variable orders. The missing ratios of variables in PAM dataset are balanced, so we didn't test it. As shown in Table 9, images from the P19 dataset have 2 empty cells at the end (6×6 grid for 34 variables), and the sorted order produces better results than random order. By contrast, the grid is full in images from the P12 dataset (6×6 grid for 36 variables), and there is no significant performance difference between these two orders, which confirms our hypothesis.

D.4 Regular Time Series Classification

Table 10: The statistics and hyperparameter settings of the evaluated regular multivariate time series datasets from the UEA Time Series Classification Archive Bagnall et al. (2018).

Datasets	Variables	Classes	Length	Train size	Grid layout	Image size	Learning rate	Epochs
EC	3	4	1,751	261	2×2	256×256	1e-4	20
UW	3	8	315	120	2×2	256×256	1e-4	100
SCP1	6	2	896	268	2×3	256×384	1e-4	100
SCP2	7	2	1,152	200	3×3	384×384	5e-5	100
JV	12	9	29	270	4×4	384×384	1e-4	100
SAD	13	10	93	6599	4×4	384×384	1e-5	20
HB	61	2	405	204	4×4	384×384	1e-4	100
FD	144	2	62	5890	12×12	384×384	5e-4	100
PS	**963**	7	144	267	32×32	384×384	5e-4	100
EW	6	5	**17984**	128	2×3	256×384	2e-5	100

The advantage of our approach is that it can be used to model any shape of time series, whether it is regular or not. We chose ten representative multivariate time series datasets from the UEA Time Series Classification Archive Bagnall et al. (2018): EthanolConcentration (EC), Handwriting (HW), UWaveGestureLibrary (UW), SelfRegulationSCP1 (SCP1), SelfRegulationSCP2 (SCP2), JapaneseVowels (JV), SpokenArabicDigits (SAD), Heartbeat (HB), FaceDetection (FD), PEMS-SF (PS), and EigenWorms (EW). As shown in Table 10, these ten datasets have diverse characteristics in terms of numbers of classes, variables, and time series length. We also present the hyperparameter and training details in the Table. It is worth mentioning that the PS dataset contains an exceptionally high number of variables (963), while the EW dataset has extremely long time series (17984). We utilize these two datasets to assess the effectiveness of our approach when dealing with large numbers of variables and long time series. We follow (Zerveas et al., 2021) to use these baselines for comparison: DTW$_D$ which stands for dimension-Dependent DTW combined with dilation-CNN (Franceschi et al., 2019), LSTM (Graves, 2012), XGBoost (Chen & Guestrin, 2016), Rocket (Dempster et al., 2020), and a transformer-based TST (Zerveas et al., 2021) which operates on fully observed numerical time series data.

The performance comparisons are shown in Table 11. Our approach performs consistently well on these datasets with different characteristics. Its average accuracy is second-best and close to the best-performed baseline method TST. By contrast, on the irregularly sampled time series datasets, ViTST outperforms Transformer (Trans-mean) as discussed in Sect 3.2. On the PS dataset, which has massive variables, and the EW dataset, which has exceptionally long time series, our approach still performs well with the same image resolution as in other datasets (384×384), indicating its effectiveness and efficiency in dealing with such time series data. It should be noted that most of the algorithms on regular and irregular time series are studied separately and could not handle the other type of time series well. However, our approach achieves promising results on both regular and irregular time series, showing its superiority in generality and effectiveness.

Table 11: Performance comparison on regular time series datasets. The EW dataset is not included in the Avg. score.

Datasets	DTW$_D$	LSTM	XGBoost	Rocket	TST	ViTST
EC	0.323	0.323	0.437	<u>0.452</u>	0.326	**0.456**
UW	0.903	0.412	0.759	**0.944**	<u>0.913</u>	0.862
SCP1	0.775	0.689	0.846	<u>0.908</u>	**0.922**	0.898
SCP2	0.539	0.466	0.489	0.533	**0.604**	<u>0.561</u>
JV	0.949	0.797	0.865	<u>0.962</u>	**0.997**	0.946
SAD	0.963	0.319	0.696	0.712	**0.998**	<u>0.985</u>
HB	0.717	0.722	0.732	0.756	**0.776**	<u>0.766</u>
FD	0.529	0.577	0.633	0.647	0.689	0.632
PS	0.711	0.399	0.983	0.751	0.896	0.913
EW	0.618	-	-	-	0.748	0.878
Avg.	0.717	0.523	0.727	0.741	**0.791**	<u>0.780</u>

D.5 Self-supervised Learning

Self-supervised learning has become a popular approach to learning representation from unlabelled data, which can benefit various downstream tasks. Masked language modeling (MLM) (Devlin et al., 2018) and masked image modeling (MIM) (Xie et al., 2022; He et al., 2022) have been dominant self-supervised approaches in NLP and CV domains. We also perform a preliminary exploration on the *masked image modeling* on time series line graph images: we mask a portion of patches in the line graph images and train the model to recover them.

As shown in Fig. 7, we randomly mask columns of patches with a width of 32 on each line graph within a grid cell. In this way, we can ensure that some regions containing line graphs are masked instead of all the empty places. The masking ratio is set as 50%. Since the self-supervised pretraining usually requires a large amount of unlabelled data, we experimented on the largest dataset P19, which has 38803 samples. We trained the Swin Transformer model for 10 epochs with batch size 48. The learning rate is 2e-5. A one-layer prediction head is applied on the vision transformer encoder to reconstruct the pixels. Following (Xie et al., 2022), we employ an ℓ_1 loss on the masked pixels:

$$\mathcal{L} = \frac{1}{\Omega(\mathbf{p}_\mathrm{M})} \left\| \hat{\mathbf{p}_\mathrm{M}} - \mathbf{p}_\mathrm{M} \right\|_1 , \tag{1}$$

where $\mathbf{p}_\mathrm{M}$ and $\hat{\mathbf{p}_\mathrm{M}}$ are the masked and reconstructed pixels, respectively; $\Omega(\cdot)$ denotes the number of elements.

After self-supervised pre-training, we further fine-tuned the model on the classification task with the same setting as directly fine-tuning: 2 epochs on the upsampled dataset with a learning rate of 2e-5. The performance improved by **1.0** AUPRC points from 52.8 ($\pm$ 3.8) to 53.8 ($\pm$ 3.2). However, AUROC points slightly dropped from 89.4 ($\pm$ 1.9) to 88.9 ($\pm$ 2.1), which is within a standard deviation. Note that we did not perform an extensive hyperparameter search in the preliminary explorations. We believe this is worth further explorations, which we leave for future work.

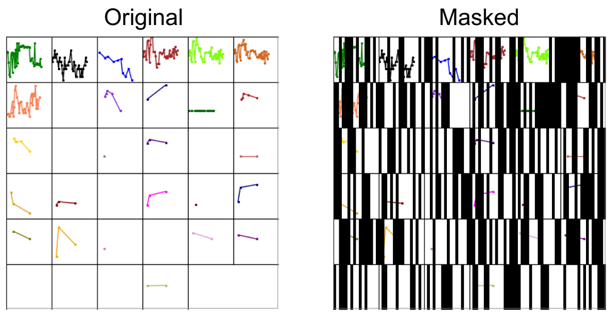


Figure 7: Illustration of the masking area with a mask ratio of 0.5 in our mask image modeling exploration on time series line graph images. The model is trained to reconstruct the masking areas.

D.6 Full Experimental Results

We presented the full experimental results in the leave-sensors-out settings in Table 12, and the full results of ablation studies on backbone vision models as illustrated in Fig. 5 are presented in Table 13.

Table 12: Full results in the leave-sensors-out settings on PAM dataset. The “missing ratio” denotes the ratio of masked variables.

Missing ratio	Methods	PAM (Leave-**fixed**-sensors-out)				PAM (Leave-**random**-sensors-out)			
		Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
10%	Transformer	60.3 ± 2.4	57.8 ± 9.3	59.8 ± 5.4	57.2 ± 8.0	60.9 ± 12.8	58.4 ± 18.4	59.1 ± 16.2	56.9 ± 18.9
	Trans-mean	60.4 ± 11.2	61.8 ± 14.9	60.2 ± 13.8	58.0 ± 15.2	62.4 ± 3.5	59.6 ± 7.2	63.7 ± 8.1	62.7 ± 6.4
	GRU-D	65.4 ± 1.7	72.6 ± 2.6	64.3 ± 5.3	63.6 ± 0.4	68.4 ± 3.7	74.2 ± 3.0	70.8 ± 4.2	72.0 ± 3.7
	SeFT	58.9 ± 2.3	62.5 ± 1.8	59.6 ± 2.6	59.6 ± 2.6	40.0 ± 1.9	40.8 ± 3.2	41.0 ± 0.7	39.9 ± 1.5
	mTAND	58.8 ± 2.7	59.5 ± 5.3	64.4 ± 2.9	61.8 ± 4.1	53.4 ± 2.0	54.8 ± 2.7	57.0 ± 1.9	55.9 ± 2.2
	Raindrop	77.2 ± 2.1	82.3 ± 1.1	78.4 ± 1.9	75.2 ± 3.1	76.7 ± 1.8	79.9 ± 1.7	77.9 ± 2.3	78.6 ± 1.8
	ViTST	**92.7** ± 0.9	**94.2** ± 0.9	**93.2** ± 0.4	**93.6** ± 0.6	**88.4** ± 1.4	**92.3** ± 0.5	**88.6** ± 1.9	**89.8** ± 1.5
20%	Transformer	63.1 ± 7.6	71.1 ± 7.1	62.2 ± 8.2	63.2 ± 8.7	62.3 ± 11.5	65.9 ± 12.7	61.4 ± 13.9	61.8 ± 15.6
	Trans-mean	61.2 ± 3.0	74.2 ± 1.8	63.5 ± 4.4	64.1 ± 4.1	56.8 ± 4.1	59.4 ± 3.4	53.2 ± 3.9	55.3 ± 3.5
	GRU-D	64.6 ± 1.8	73.3 ± 3.6	63.5 ± 4.6	64.8 ± 3.6	64.8 ± 0.4	69.8 ± 0.8	65.8 ± 0.5	67.2 ± 0.0
	SeFT	35.7 ± 0.5	42.1 ± 4.8	38.1 ± 1.3	35.0 ± 2.2	34.2 ± 2.8	34.9 ± 5.2	34.6 ± 2.1	33.3 ± 2.7
	mTAND	33.2 ± 5.0	36.9 ± 3.7	37.7 ± 3.7	37.3 ± 3.4	45.6 ± 1.6	49.2 ± 2.1	49.0 ± 1.6	49.0 ± 1.0
	Raindrop	66.5 ± 4.0	72.0 ± 3.9	67.9 ± 5.8	65.1 ± 7.0	71.3 ± 2.5	75.8 ± 2.2	72.5 ± 2.0	73.4 ± 2.1
	ViTST	**88.4** ± 1.0	**90.4** ± 1.4	**89.3** ± 0.8	**89.7** ± 1.0	**85.1** ± 1.2	**91.1** ± 1.0	**85.6** ± 1.0	**87.0** ± 1.0
30%	Transformer	31.6 ± 10.0	26.4 ± 9.7	24.0 ± 10.0	19.0 ± 12.8	52.0 ± 11.9	55.2 ± 15.3	50.1 ± 13.3	48.4 ± 18.2
	Trans-mean	42.5 ± 8.6	45.3 ± 9.6	37.0 ± 7.9	33.9 ± 8.2	65.1 ± 1.9	63.8 ± 1.2	67.9 ± 1.8	64.9 ± 1.7
	GRU-D	45.1 ± 2.9	51.7 ± 6.2	42.1 ± 6.6	47.2 ± 3.9	58.0 ± 2.0	63.2 ± 1.7	58.2 ± 3.1	59.3 ± 3.5
	SeFT	32.7 ± 2.3	27.9 ± 2.4	34.5 ± 3.0	28.0 ± 1.4	31.7 ± 1.5	31.0 ± 2.7	32.0 ± 1.2	28.0 ± 1.6
	mTAND	27.5 ± 4.5	31.2 ± 7.3	30.6 ± 4.0	30.8 ± 5.6	34.7 ± 5.5	43.4 ± 4.0	36.3 ± 4.7	39.5 ± 4.4
	Raindrop	52.4 ± 2.8	60.9 ± 3.8	51.3 ± 7.1	48.4 ± 1.8	60.3 ± 3.5	68.1 ± 3.1	60.3 ± 3.6	61.9 ± 3.9
	ViTST	**84.1** ± 1.3	**86.5** ± 0.4	**83.1** ± 0.8	**84.9** ± 1.0	**80.6** ± 1.2	**89.5** ± 1.3	**80.9** ± 1.1	**82.6** ± 1.1
40%	Transformer	23.0 ± 3.5	7.4 ± 6.0	14.5 ± 2.6	6.9 ± 2.6	43.8 ± 14.0	44.6 ± 23.0	40.5 ± 15.9	40.2 ± 20.1
	Trans-mean	25.7 ± 2.5	9.1 ± 2.3	18.5 ± 1.4	9.9 ± 1.1	48.7 ± 2.7	55.8 ± 2.6	54.2 ± 3.0	55.1 ± 2.9
	GRU-D	46.4 ± 2.5	64.5 ± 6.8	42.6 ± 7.4	44.3 ± 7.9	47.7 ± 1.4	63.4 ± 1.6	44.5 ± 0.5	47.5 ± 0.0
	SeFT	26.3 ± 0.9	29.9 ± 4.5	27.3 ± 1.6	22.3 ± 1.9	26.8 ± 2.6	24.1 ± 3.4	28.0 ± 1.2	23.3 ± 3.0
	mTAND	19.4 ± 4.5	15.1 ± 4.4	20.2 ± 3.8	17.0 ± 3.4	23.7 ± 1.0	33.9 ± 6.5	26.4 ± 1.6	29.3 ± 1.9
	Raindrop	52.5 ± 3.7	53.4 ± 5.6	48.6 ± 1.9	44.7 ± 3.4	57.0 ± 3.1	65.4 ± 2.7	56.7 ± 3.1	58.9 ± 2.5
	ViTST	**76.5** ± 1.9	**83.5** ± 0.9	**76.7** ± 2.4	**78.3** ± 2.1	**73.7** ± 2.2	**86.4** ± 1.1	**74.0** ± 2.2	**75.8** ± 1.8
50%	Transformer	21.4 ± 1.8	2.7 ± 0.2	12.5 ± 0.4	4.4 ± 0.3	43.2 ± 2.5	52.0 ± 2.5	36.9 ± 3.1	41.9 ± 3.2
	Trans-mean	21.3 ± 1.6	2.8 ± 0.4	12.5 ± 0.7	4.6 ± 0.2	46.4 ± 1.4	59.1 ± 3.2	43.1 ± 2.2	46.5 ± 3.1
	GRU-D	37.3 ± 2.7	29.6 ± 5.9	32.8 ± 4.6	26.6 ± 5.9	49.7 ± 1.2	52.4 ± 0.3	42.5 ± 1.7	47.5 ± 1.2
	SeFT	24.7 ± 1.7	15.9 ± 2.7	25.3 ± 2.6	18.2 ± 2.4	26.4 ± 1.4	23.0 ± 2.9	27.5 ± 0.4	23.5 ± 1.8
	mTAND	16.9 ± 3.1	12.6 ± 5.5	17.0 ± 1.6	13.9 ± 4.0	20.9 ± 3.1	35.1 ± 6.1	23.0 ± 3.2	27.7 ± 3.9
	Raindrop	46.6 ± 2.6	44.5 ± 2.6	42.4 ± 3.9	38.0 ± 4.0	47.2 ± 4.4	59.4 ± 3.9	44.8 ± 5.3	47.6 ± 5.2
	ViTST	**70.0** ± 2.7	**79.9** ± 2.2	**70.5** ± 3.1	**72.2** ± 3.0	**70.9** ± 1.2	**83.6** ± 2.4	**71.5** ± 1.4	**73.3** ± 2.1

Table 13: Full results of our approach with different backbone vision models and the compared baselines. **Bold** indicates the best performer, while underline represents the second best.

Methods	P19		P12		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1 score
Transformer	80.7 ± 3.8	42.7 ± 7.7	83.3 ± 0.7	47.9 ± 3.6	83.5 ± 1.5	84.8 ± 1.5	86.0 ± 1.2	85.0 ± 1.3
Trans-mean	83.7 ± 1.8	45.8 ± 3.2	82.6 ± 2.0	46.3 ± 4.0	83.7 ± 2.3	84.9 ± 2.6	86.4 ± 2.1	85.1 ± 2.4
GRU-D	83.9 ±1.7	46.9 ± 2.1	81.9 ± 2.1	46.1 ± 4.7	83.3 ± 1.6	84.6 ± 1.2	85.2 ± 1.6	84.8 ± 1.2
SeFT	81.2 ± 2.3	41.9 ± 3.1	73.9 ± 2.5	31.1 ± 4.1	67.1 ± 2.2	70.0 ± 2.4	68.2 ± 1.5	68.5 ± 1.8
mTAND	84.4 ± 1.3	50.6 ± 2.0	84.2 ± 0.8	48.2 ± 3.4	74.6 ± 4.3	74.3 ± 4.0	79.5 ± 2.8	76.8 ± 3.4
IP-Net	84.6 ± 1.3	38.1 ± 3.7	82.6 ± 1.4	47.6 ± 3.1	74.3 ± 3.8	75.6 ± 2.1	77.9 ± 2.2	76.6 ± 2.8
DGM^2-O	86.7 ± 3.4	44.7 ± 11.7	84.4 ± 1.6	47.3 ± 3.6	82.4 ± 2.3	85.2 ± 1.2	83.9 ± 2.3	84.3 ± 1.8
MTGNN	81.9 ± 6.2	39.9 ± 8.9	74.4 ± 6.7	35.5 ± 6.0	83.4 ± 1.9	85.2 ± 1.7	86.1 ± 1.9	85.9 ± 2.4
Raindrop	87.0 ± 2.3	51.8 ± 5.5	82.8 ± 1.7	44.0 ± 3.0	88.5 ± 1.5	89.9 ± 1.5	89.9 ± 1.5	89.8 ± 1.0
ResNet	76.3 ± 3.3	34.7 ± 4.1	71.9 ± 1.0	28.8 ± 2.4	73.1 ± 0.9	82.4 ± 5.6	69.7 ± 0.9	71.4 ± 1.8
ViT	87.9 ± 2.5	51.6 ± 3.7	84.8 ± 1.3	48.1 ± 3.8	93.4 ± 0.7	94.7 ± 0.9	94.1 ± 0.7	94.3 ± 0.7
Swin	**89.4** ± 1.9	**52.8** ± 3.8	**85.6** ± 1.1	**49.8** ± 2.5	**96.1** ± 0.7	**96.8** ± 1.1	**96.5** ± 0.7	**96.6** ± 0.9
Swin-scratch	74.6 ± 2.5	29.9 ± 4.6	66.9 ± 1.6	26.5 ± 2.6	84.5 ± 0.5	86.6 ± 0.6	87.1 ± 1.2	86.6 ± 0.6